CSCI–572 Assignment 1

Crawling using Apache Nutch

Dark and deep web are two potential places to sell guns illegally. We used Apache Nutch to crawl for images of guns, Apache Tika for content and metadata extraction, and Selenium to automate Ajax interaction and more.

Team #32

Sanjay Jagaeesh

Manohar Thagadur Natraju

Soumya Gowda

Pramod P. Setlur

CSCI–572 Assignment 1

Question 1: Download and configure Nutch to crawl Weapons images as identified in the seed list.

a. The list of files we modified to deal with politeness are as follows:

Nutch-Site.xml:

1. We set http.agent.rotate to 'true' and added 7 agent names in agents.txt.
2. We increased the http.timeout from 10,000 to 80,000.
3. We changed the http.content.limit to -1 to remove the limitation on the size of the crawling file.
4. We increased the fetcher.threads.fetch from 10 to 20.
   Our Nutch-Site.xml contains the other changes.

b.

i. Regex-urlfilter.txt and Automaton-urlfilter.txt
   1. We skipped domains like facebook.com, tumblr.com, twitter.com, etc. to filter majority of images unwanted to us.
   2. We removed the image extensions to avoid skipping images.

ii. Mimetype-filter.txt:
   1. We added image/* mimetype to allow crawling of documents with images/* and text/html mimetype.

iii. Suffix-urlfilter.txt:
   1. We commented all the image suffixes to prevent them from exclusion.

c. We enabled rotating bots in agents.txt.
   i. Spider#32Crawler, MasosapaCrawler, etc

Question 2: Perform crawls of Weapon's images sites

a. We use the NutchPy library
b. We wrote a python script to generate all the image mimetypes that were found in our crawled data.
c. We found the following mime types:
   1. image/png
   2. image/jpg
   3. image/jpeg
   4. image/gif
   5. image/bmp
   6. image/x-icon
   7. image/tiff
   8. image/x-xbitmap
   9. image/pjpeg

Many of the URLs that weren't fetched were mainly because of java.net.SocketTimeoutException, java.lag.IllegalArguementExeption, HTTP 403.

d. The list of 100 Urls that we found difficulty in fetching has been added in "*URLs not fetched.xlsx*"
e. We crawled all the 85 urls with 30 rounds
The crawl stats at this stage is tabulated as follows:-

$ bin/crawl urls/ ~/CSCI-572/guns_crawl_Step2 30
$ bin/nutch readdb ~/CSCI-572/guns_crawl_Step5/crawldb -stats

Total URLS	4710184
retry 0:	4625184
retry 1:	64302
retry 2:	20698
Min Score	0.0
Avg Score	2.9849576E-5
Max Score	1.173
Status 1 (db_unfetched)	4348772
Status 2 (db_fetched)	314453
Status 3 (db_gone)	28243
Status 4 (db_redir_temp)	5114
Status 5 (db_redir_perm)	6146
Status 6 (db_notmodified)	7
Status 7 (db_duplicate)	7449

Question 3: Use the information you learn in question 2 to extend the Nutch Selenium Protocol Plugin

a. We used the protocol-interactive-plugin for Nutch and enabled it go past the pages that require Ajax interaction.
b. We wrote *Custom Handlers* trying to generalize a pattern for the websites that required a login or those which were paginated.
c. Upon building and testing Nutch with the plugin enabled we could see Firefox pop open and see Nutch attempting to follow the instructions we had written in our *Custom Handlers*.

Question 4: Build the latest version of Tika and install Tesseract

a, b, c. We installed Tesseract and upgraded Tika. Thirebuild Nutch

Question 5: Re-run your Weapons Crawls with enhanced Tika and Nutch Selenium you have built in step 3

a. The list of 100 Urls that we found difficulty in fetching has been added in "*URLs after Selenium.xlsx*"
b. About 50% of the Urls from 2d are still present in the result of this step. The exact urls can be found in the above excel sheet.

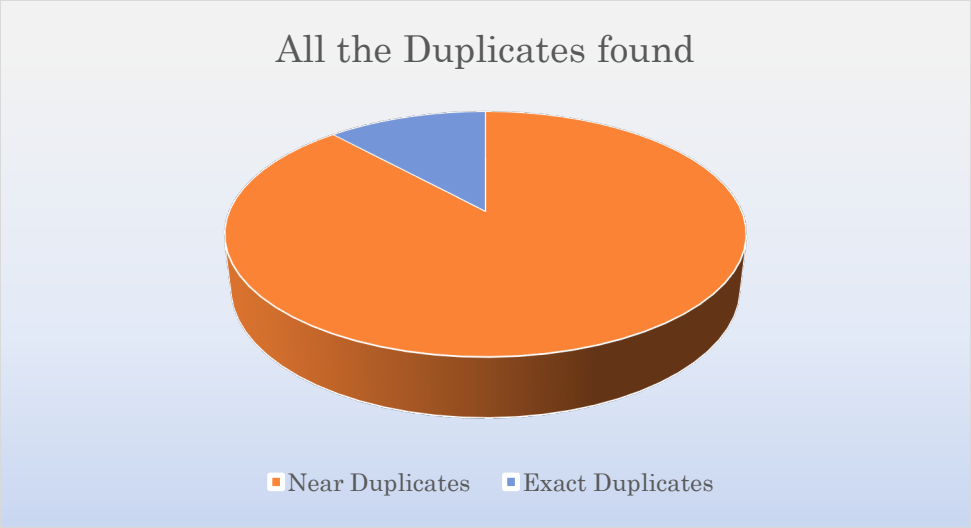
c. The un-fetched URLs from Nutch crawls were not fetched for the main reason that corresponding mime types couldn't be parsed by Nutch. Tika parsing enabled to overcome this hindrance and parse additional mime type related to images.
d. The crawl stats at this stage is tabulated as follows:-

$ bin/crawl urls/ ~/CSCI-572/guns_crawl_Step5 30
$ bin/nutch readdb ~/CSCI-572/guns_crawl_Step5/crawldb -stats

Total URLS	4067987
retry 0:	3898790
retry 1:	35875
retry 2:	9765
Min Score	0.0
Avg Score	2.3457E-3
Max Score	1.173
Status 1 (db_unfetched)	3267867
Status 2 (db_fetched)	340989
Status 3 (db_gone)	34075
Status 4 (db_redir_temp)	4226
Status 5 (db_redir_perm)	8632
Status 6 (db_notmodified)	11
Status 7 (db_duplicate)	6508

Question 6: Develop two deduplications to use on the extracted text and meta data in the parsed content from Nutch

a. We developed an algorithm to find exact duplicates. Here is the gist of the algorithm:
   - Read the dump file generated after reading the segments.
   - Filter this file with lines containing only JPGs, PNGs and GIFs.
   - Using *tika-python* extract the meta data for the above lines and output this into another file for further processing.
   - Read the above file that contains the *parse text* for every image. Extract every fifth character to create a fingerprint.
   - Hash the above fingerprint using MD5 and add all the duplicates to a dictionary.
b. We developed an algorithm to find near duplicates using Simhash:
   -



Question 7: Enable the Nutch similarity scoring filter focused crawling plugin

a. We went through the documentation of Similarity Scoring Filter and understood the main purpose of the scoring filter.
b. We enabled this plugin and re-ran the weapon crawls.
c. We added the goldstandard.txt with all words that were relevant to weapons. And, we added the stop words that are meant for inverse document frequency inside the stopwords.txt. We could see improvements in the results of near deduplication algorithms.
d. The crawl stats at this stage is tabulated as follows:-

```
$ bin/crawl urls/ ~/CSCI-572/guns_crawl_Step7 30
$ bin/nutch readdb ~/CSCI-572/guns_crawl_Step7/crawldb -stats
```

Total URLS	3568756
retry 0:	25687
retry 1:	8643
retry 2:	6489
Min Score	0
Avg Score	2.36387E-3
Max Score	1.932
Status 1 (db_unfetched)	2878690
Status 2 (db_fetched)	45876
Status 3 (db_gone)	5435
Status 4 (db_redir_temp)	7456
Status 5 (db_redir_perm)	4563
Status 6 (db_notmodified)	86
Status 7 (db_duplicate)	12564

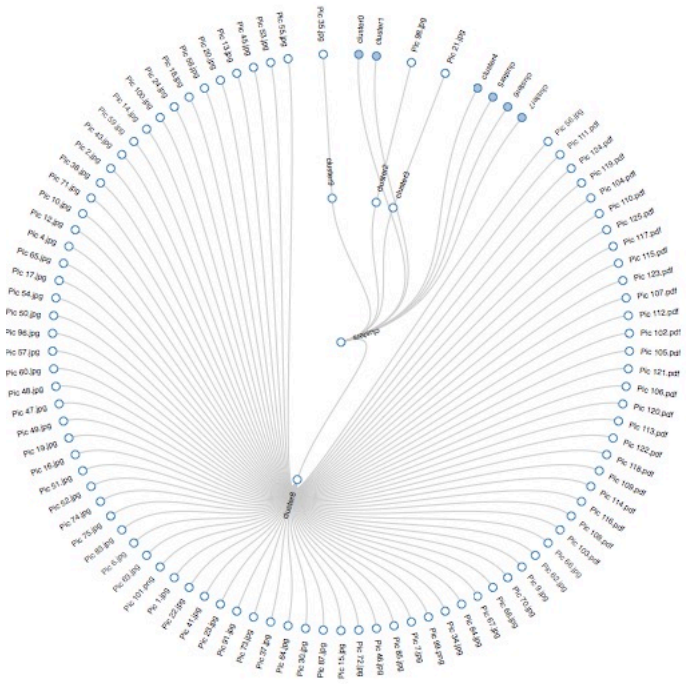
EXTRA CREDITS

Question 8: Download and configure Nutch python which you will use to control your crawls

a. We went over the documentation and wrote a python program to crawl with Nutch using Nutch rest server and Nutch python.
b. The python program *Nutch_python_crawl.py* is attached.
c. We went through the documentation and implemented the best practices for crawling using Nutch rest server.
d. NutchPy is a python library for working with Apache Nutch. It has functionality to work with Nutch data structures – to read Sequence files, LinkDb, etc. We feel these functionalities can be added inside the *Nutch REST server* as an API. It can accept a link to the the segments generated and further processing of can be performed by the *Nutch REST server*.

Question 9: Dump the crawl data out of your Nutch content and then run tika-simlarity over it

a. Here is what tika-similarity inferred with all the images that Nutch was able to crawl successfully.



b. We could see an interesting pattern in the D3 that was generated by *tika-similarity*. Many images that were actually guns followed a noticeable pattern.

c. Cluster 0 consisted of about 30,000 images many of which were guns. Cluster 1 contains images of that can be excluded from guns.